Int. J. Communications, Network and System Sciences, **2012, 5, 286-297**
http://dx.doi.org/10.4236/ijcns.2012.55038 Published Online May 2012 (http://www.SciRP.org/journal/ijcns)

Advanced Transition/Cluster Key Management Scheme for End-System Multicast Protocol

Ayman El-Sayed
Department of Computer Science and Engineering, Faculty of Electronic Engineering, Menouf, Egypt
Email: ayman.elsayed@el-eng.menofia.edu.eg

Received March 2, 2012; revised March 23, 2012; accepted March 31, 2012

ABSTRACT

The recent growth of the World Wide Web has sparked new research into using the Internet for novel types of group communication, like multiparty videoconferencing and real-time streaming. Multicast has the potential to be very useful, but it suffers from many problems like security. To achieve secure multicast communications with the dynamic aspect of group applications due to free membership joins and leaves in addition to member's mobility, key management is one of the most critical problems. So far, a lot of multicast key management schemes have been proposed and most of them are centralized, which have the problem of "one point failure" and that the group controller is the bottleneck of the group. In order to solve these two problems, we propose a Key Management Scheme, using cluster-based End-System Multicast (ESM). The group management is between both 1) the main controller (MRP, Main Rendezvous Point) and the second controllers (CRP, Cluster RP); and 2) the second controllers (CRPs) and its members. So, ESM simplifies the implementation of group communication and is efficient ways to deliver a secure message to a group of recipients in a network as a practical alternative to overcome the difficulty of large scale deployment of traditional IP multicast. In this paper, we analyze different key management schemes and propose a new scheme, namely Advanced Transition/Cluster Key management Scheme (ATCKS) and find it has appropriate performance in security.

Keywords: Multicast Protocol; End-System Multicast; Application-Level Multicast; Security; Group Key Management

1. Introduction

End-System Multicast (ESM) [1] aims to simplify group communication by implementing the multicast functionality at the application layer instead of the networking layer. Using ESM, an end host can join/leave a multicast group without the need of native multicast support at the network routers. ESM protocol constructs and maintains an overlay tree between the end host members of the multicast group, and the multicast data traffic is transmitted from one member to another one via this overlay tree using unicast connections. This gives the following advantages for ESM solutions compared to the classical IP multicast. 1) ESM is easy to deploy. It does not require changes at the network layer and it does not require universal network support to be deployed; 2) The construction of a logical overlay topology hides routing complications such as link failure instances; 3) Intermediate nodes do not have to maintain per group state for each multicast group; 4) In the ESM architecture, all multicast related tasks such as group membership management, packet replication and forwarding and state maintenance are handled by the end hosts. Intermediate routers play no role in an ESM protocol. Finally, ESM can exploit the capabilities of lower layer protocols to provide reliability, congestion control, flow control or security according to the needs of the application. End-System Multicast (ESM) [1,2] protocols creates an overlay depending on the different metrics between the group members such as the Round Trip Time (RTT) and loss rate. Each group member collects the metrics between itself and the other group members. The End-System Multicast (ESM) can be classified into two main categories: 1) tree first approaches [3]; 2) Mesh first approaches, like NARADA [4], centralized protocol as both ESM [5] and Host-based Multicast (HBM) [6], and semi centralized ESM as described in [1,7]. In [8], a highly scalable locating algorithm is proposed to gradually direct newcomers to a set of their closest nodes without inducing high overhead. On the basis of this locating process, they build a robust and scalable topology-aware clustered hierarchical overlay scheme to support large scale multicast applications. One of security issues in ESM is the problem of cheating behavior in ESM session. Some of group members may be cheats. A cheat informs RP that it has the minimum delay to the source and the infinity delay to the other group members. The overlay multicast is not efficient due to the wrong metrics between group members because of these cheats. Authors in [9] analyzed the negative

impact of cheating about the distance metric on the link stress and on the link stretch of the constructed overlay tree. Authors in [10] proved the negative impact of cheating on the stability of the overlay tree topology. Authors in [11] analyzed the negative impact of cheating nodes on the performances and on the stability of overlay tree constructed by the protocol MDA-ALM [12]. Authors in [13] studied how to detect the presence of malicious activities in the overlay tree based on their history performance. Each member monitors the performances of its neighbor(s) and reports them to the RP. Based on these reports, the RP calculates members reputation, and considers as malicious nodes members having low reputation. Author in [14] proposed detecting/avoiding cheating algorithms. Because of these algorithms, the cheating effect is canceled and the overlay topology with cheating likes the overlay topology without cheating. Despite all these advantages, ESM do not simplify security issues for group communications compared to the traditional IP Multicast and an important research effort should be dedicated to this issue in order to propose robust and viable ESM solutions. These security services can be facilitated if group members share a common secret, which in turn makes key management a fundamental challenge in designing secure multicast and reliable group communication systems. Many solutions have been proposed in the multicast literature to tackle the problem of key management within flat groups [15]. The Key management includes creating, distributing and updating the keys then it constitutes a basic block for secure group communication applications. One of the primary objectives of any key management scheme is the secure distribution of keying material. Most of the security services rely generally on encryption using Traffic Encryption Keys (TEKs). Each member holds a key to encrypt and decrypt the multicast data. When a member joins and leaves a group, the key has to be updated and distributed to all group members in order to meet the above requirements. The process of updating the keys and distributing them to the group members is called rekeying operation [16]. Rekeying is required in secure multicast communication to ensure that a new member cannot decrypt the stored multicast data (before its joining) and prevents a leaving member from eavesdropping future multicast data. The rekey process should be done after each membership change, and if the membership changes are frequent, key distribution will require a large number of key exchanges per unit time in order to maintain both forward and backward secrecies. The number of TEK update messages in the case of frequent join and leave operations induces high packet loss rates and reduces key delivery ratio which makes unreliable. In this paper, we propose a new key management scheme, called Advanced Transition/ Cluster Key Scheme (ATCKS), to be used in a centralized ESM protocol. ATCKS reduces the key management overhead by using a unique TEK for each cluster, and a set of transition keys for members who recently joined the cluster. ATCKS is periodically updated in order to integrate members who are in the transition state for each cluster (managed by individual transition keys) with its cluster. The use of a unique key for each cluster both decreases the key updating overheads and avoids important decryption and re-encryption overheads, and the use of a small set of transition keys for each cluster, decreases the key updating overhead as well. The remainder of the paper is organized as follows: a centralized End-system multicast is described in Section 2. A group key management is presented in Section 3. In Section 4, we detail our new key management scheme. Key management and data encryption/decryption overhead analysis of our ATCKS are presented in Section 5. The results of performance of our scheme are compared to other schemes, in Section 6. Finally, the paper is concluded in Section 7.

2. A Centralized ESM Protocol

In the Centralized ESM [1], everything is under the control of a single host, the Main Rendezvous Point (or MRP). MRP will not be connected to the members but connected to CRPs that are connected to the members. The MRP knows the members, their features, and the communication costs between them. It is responsible of the overlay topology creation. The CRP knows subset of the members, their features, and the communication costs between each member of this subset and all members in the group. It is responsible of collecting the communication costs and informing MRP. Each member evaluates the communication cost between itself and all the members and informs its CRP. The flow of control traffic is both between MRP and CRPs, and between a CRP and its members. For example of N (e.g. 90) members split into three subsets (clusters) of K (e.g. 30) members and CRP for each cluster. The whole overlay topology is created by MRP and MRP informs each CRP by the neighbors of each member in its cluster. The data flow among group members that are connected as cluster-based, so each cluster is controlled by a CRP. There are some members, namely Gate Way Member (GWM), in each cluster used to interconnect the clusters to each other. GWM is a member controlled by MRP via CRPs of the clusters in which this member inside.

3. Group Key Management

In order to secure group communications, security mechanisms such as authentication, access control, integrity verification and group confidentiality are required. Among which group confidentiality is the most important service

for several applications like military and fire brigades. Most of these mechanisms rely generally on encryption using one or several Traffic Encryption Keys (TEKs) [17]. TEK is a symmetric key that is used to encrypt messages. TEKs are typically changed frequently, in some systems daily and in others for every message, or as well as membership change in the multicast group. The management of these keys, which includes creating, distributing and updating the keys, constitutes then a basic block to build secure group communication applications. Group communication confidentiality requires that only valid users could decrypt the multicast data sent on the overlay tree. The main goal of a group key management is to reduce the impact of membership changes while ensuring the security requirements during the multicast session. Due to resource constraints secure and efficient group communication in mobile ad hoc networks is a very challenging task, therefore, any proposed group key establishment protocol must be efficient in computation and communication. In [18,19], the authors present a secure and efficient group key agreement protocol for mobile ad hoc network.

Group Key management in ESM has received less attention. Authors in [20] classified key management schemes for Overlay Multicast into three categories: Group Key scheme (GKS), Neighbors Key scheme (NKS), and Clusters Key scheme (CKS). In the Group Key Scheme, a unique TEK is shared among all hosts and is used to encrypt and decrypt all the needed traffic. In the Neighbors Key Scheme, each pair of neighbors in the overlay share a specific key, packets are decrypted and re-encrypted at each intermediate host along each path in the tree. In the Clusters Key Scheme, which can be applied in case of hierarchical ESM overlays, a separate key is shared among all members of a given cluster, encryption and decryption of the traffic is done only at the frontiers between the clusters. There is another scheme is proposed in [17] called Transition key scheme (TKS), each message is encrypted/decrypted once for all members in the group, except for those who are still in the transition state. For those specific members the message is decrypted/re-encrypted once again by their parents before it is forwarded through the transition secured channel. Another proposed scheme is integrated between CKS and TKS, namely Transition cluster Key Scheme (TCKS) [21]. One of five proposed schemes in [17], namely NKS, has higher overhead than GKS, CKS, TKS, and TCKS, so in this paper, our scheme is compared with GKS, CKS, TKS, and TCKS only. **Figure 1** gives an overview of these schemes as well as our scheme.

3.1. Group Key Scheme (GKS)

In the Group Key Scheme (GKS) scheme, a unique secret key is shared among all the ESM group members. The MRP should update this TEK after each membership change event. During a join process, a new member establishes a secure channel with the MRP and asks for a new TEK. The MRP triggers the rekeying process, generates a new TEK and sends this new TEK to the new member. The MRP may use the old TEK to securely send the new TEK to all other members of the group. After a leave event, the MRP is notified and then it triggers a rekeying process and generates a new TEK that it will send to remaining members through individual secure channels. Using this scheme, data is encrypted once at the MRP and decrypted at end-host members; no intermediate decryption/re-encryption operations are needed.

3.2. Clusters Key Scheme (CKS)

Some ESM protocols use hierarchical clusters to construct the overlay tree, called clustered ESM protocol [1]. The Clusters Key Scheme (CKS) [2] may be seen as a balance between the GKS and NGS schemes. In this case we can use a separate TEK for each cluster. As a joining/leaving member is assigned in the layer zero of the hierarchical. When a join or leave event occurs within a given cluster, the cluster leader (CRP) triggers a local TEK rekeying which will affect only members in the concerned cluster. To traverse from one cluster to another, the group traffic should be decrypted and re-encrypted at

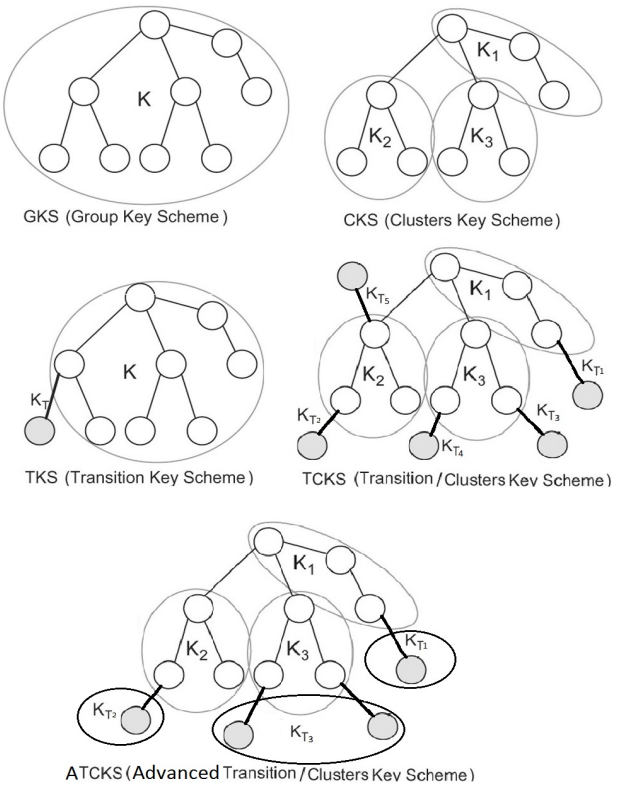

Figure 1. ESM Key Management Schemes.

the clusters boundaries. This scheme reduces the rekeying overhead and keeps the decryption/re-encryption overhead acceptable. Here each message is encrypted once at the source and decrypted/re-encrypted by the GWM at the boundary of adjacent clusters.

3.3. Transition Key Scheme (TKS)

In the TKS scheme, a unique TEK is used for the group. This TEK is updated periodically, as in all other schemes, in order to prevent cryptanalysis attacks. During one TEK period, if a new member joins the group, it will establish a separate secure channel with its parent using an individual TEK, called Transition Key. Using this way, there is no need to update the TEK for other members in the session. This new member will stay in this transition state until the next rekeying period; it will then receive the new group TEK and will no longer use a separate channel. After a leave event, a TEK renewal is always needed to ensure forward confidentiality. In TKS scheme, each message is encrypted/decrypted once for all members in the group, except for those who are still in the transition state. For those specific members the message is decrypted/re-encrypted once again by their parents before it is forwarded through the transition secured channel.

3.4. Transition/Cluster Key Scheme (TCKS)

In the TCKS scheme, we propose to use a separate TEK for each cluster and this TEK is updated periodically. During one TEK period of a cluster, if a new member joins the cluster via its CRP by aiding the MRP, it will establish a separate secure channel with its parent using an individual TEK, called Transition Key. This scheme is noted as the integration of both CKS and TKS. During a join process, there is no need to update the TEK for other members in the cluster because the new member use individual channel to connect its parent. After a leave event, a TEK renewal in its cluster only is always needed to ensure forward confidentiality. This makes the rekeying process overhead less than that of TKS. In TCKS scheme, each message is encrypted/decrypted at the boundary of clusters, depending on number of clusters, and at members who are still in the transition state. The message is decrypted/re-encrypted once again by the parents of members in the transition state.

3.5. Advanced Transition/Cluster Key Scheme (ATCKS)

In the ATCKS scheme, we use a separate TEK for each cluster and this TEK is updated periodically. During one TEK period of a cluster, if a new member joins the cluster via its CRP by aiding the MRP, it will establish a separate secure channel with its parent using an individual TEK, called Transition Key for this cluster. This scheme is noted as the integration of CKS, TKS, and TCKS. During a join process, there is no need to update the TEK for other members in the cluster because the new member use individual channel to connect its parent by using cluster's transition TEK that is given from its CRP. After a leave event, a TEK renewal in its cluster only is always needed to ensure forward confidentiality. This makes the rekeying process overhead less than that of TKS and TCKS. In ATCKS scheme, each message is encrypted/ decrypted at the boundary of clusters, depending on number of clusters, and at members who are still in the transition state. The message is decrypted/re-encrypted once again by the parents of members in the transition state.

4. A Secure Group Management in ESM

4.1. Secure Multicast Session Initialization

In the overlay multicast session, encrypting the data using a key is shared by all legitimate group members. The MRP plays the role of CRP key management controller, while the CRPs play the role of cluster key management controller. The MRP maintains all the public keys of both CRPs and legitimate members in the overlay groups. To initialize the secure multicast session, the MRP generates a new TEK and sends it to all CRPs, and CRP generates a new TEK and send it to all cluster members, this process is called secure multicast session Initialization process as shown in the following:

Secure Multicast Session Initialization Process

Step 1: *MRP generates the key* (TEK_{rp}) *in between clusters*
Step 2: *For all CRPs* $CRP_i \in CRPs$
MRP Sends $\{TEK_{cc}\}$ $Pub - CRP_i$ *to* CRP_i
Endfor
Step 3: *For all CRPs* $CRP_i \in CRPs$
CRP_i *generates the key* (TEK_i)
For all Member $m \in$ *members in cluster* (i)
CRP_i *Sends* $\{TEK_i\}$ $Pub - m$ *to* m
Endfor
Endfor

4.2. Joining a Group

The MRP address is supposed to be known by all potential members. A new member joins a group by sending join request to the MRP, and then MRP sends a list of CRPs. This new member evaluates the communication cost between itself and each CRP. This new member sends join request to the CRP that the MRP selects, and then the MRP informs both the new member and the closed CRP. The closed CRP attaches this new member to its cluster. First, the join process take place between new member, MRP, and CRP that is selected by MRP.

Figure 2 details the joining scenario between both MRP and closed CRP and a new member as follows:

1) The new member sends a *Join_request* message to the MRP.
2) MRP responds with a *certificate_request* message.
3) The new member responds by sending its certificate.
4) The MRP authenticates the new member and verifies if it is authorized to access the overlay session. The simple list of authorized members (*i.e.* members and CRPs) Access Control List (ACL) in the session is sent by MRP to all members at the beginning of the session, or through a dedicated Authentication, Authorization, and Accounting (AAA) server used by MRP.
5) In case of successful authentication and authorization, the MRP sends *join_accept* message to the new member.
6) Also, in case of successful authentication and authorization, the MRP sends *Inform_newMem* message to a CRP that is selected by MRP as a controller of the new member depending upon the communication cost between the new member and each CRP.
7) The new member also requests the certificate of CRP to accomplish the mutual authentication.
8) The CRP then sends its certificate.
9) Whence the mutual authentication is done.
10) The new member send *Join_ok* message to CRP.

After the successful authentication and authorization between the new member and its CRP is realized. The joining scenario between a new member and its parent in the cluster then is needed. **Figure 3** details steps in the join process between new member and its parent as follows:

1) The new member sends a Join_request message to the Parent that is selected by CRP.
2) The parent responds with a certificate_request message to the new member.

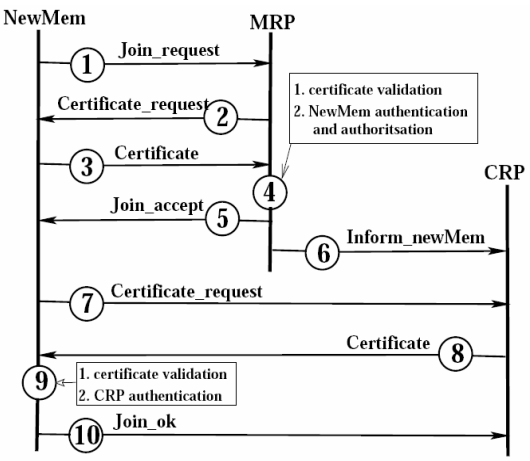

Figure 2. New member join a multicast group.

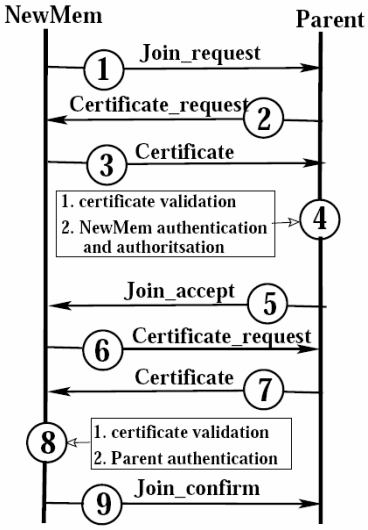

Figure 3. New member join its parent in a cluster.

3) The new member responds by sending its certificates.
4) Whence the mutual authentication is done.
5) In case of successful authentication and authorization, the parent sends join_accept message to the new member.
6) After the new member receives join_accept message, it then sends certificate_request message to the parent.
7) The parent sends its certificate.
8) Whence the mutual authentication is done.
9) The new member and its parent trigger a key agreement process to establish a temporary Transition Key to be used by the parent to encrypt the forwarded data.

The TCKS join process is illustrated as follows:

A new member joins the group (Join Process)

Let NewMem be the new joining member to the group and P be the selected parent for NewMem in the cluster C.

Step 1: *P generates a transition key* TEK_t *where* $TEK_t \neq TEK$ *of cluster C.*

Step 2: *P Sends* $\{TEK_t\}$ *Pub − NewMem to NewMem.*

4.3. Leaving a Group

A group member leaving the session gracefully informs its CRP by sending a Leave_request message, and then waits until it receives a Fwd_Leave_accept message. In the meantime if the neighbors of this leaving member are in the same cluster, then the CRP creates a new sub-topology among the neighbors of the leaving member, and informs both these neighbors and MRP. The CRP informs everybody in its cluster that a member has left the session.

Figure 4 details the leave scenario in case that neighbors of leaved member are in the same cluster, as follows:

1) The leaved member sends a Leave_request message to the CRP.
2) The CRP then sends a Leave_inform message to the MRP.
3) The MRP responds by Leave_accept message.
4) Whence the CRP receives Leave_accept message, it sends FwdLeave_accept to leaved member.
5) If the leaved member in Group Member in Cluster (GMC), the CRP generates new TEK and then sends to the member in cluster.

Otherwise, if one of neighbors is outside the cluster, then CRP forwards the leave message to the MRP to create a new sub-topology among the neighbors of the leaving member, and informs all (e.g. CRP of this clusters as well). CRP sends the *Fwd_Leave_accept* message to leaving member.

Figure 5 details the leave scenario in case those neighbors of leaved member are in the different clusters, as follows:

1) The leaved member sends a Leave_request message to the CRP.
2) The CRP then sends a Leave_inform message to the MRP.
3) The MRP responds by Leave_accept message.
4) Whence the CRP receives Leave_accept message, it sends FwdLeave_accept to leaved member.
5) The MRP sends New TEKc message to ask CRPs of the neighbors to change the key TEK.
6) If the leaved member in Group Member in Cluster (GMC), the CRP generates new TEK and then sends to the member in cluster.
7) The CRPs of neighbors generates new TEK and then sends to the member in its cluster.

The TCKS leave process is illustrated as follows:

A leaving mem. leaves the group (Leave Process)

Let LeaveMem be the leaving member from the group and P be the parent of LeaveMem in the cluster C that having a controller CRPc. Let GMC be the set of all members in cluster C except members in transition state. Let GMCT be all the member in cluster C that are in the transition state. Suppose the current TEK in the Cluster C is TEK_i.

Steps: *If* (*LeaveMem* ∈ *GMCT*) *then*
P close the secured channel with LeaveMem.
Endif
If (*LeaveMem* ∈ *GMC*) *then*
For All Clusters of LeaveMem c,
CRPc generates a new key TEK:TEK_{i+1} *for cluster c.*
For all member m ∈ *cluster c* (*GMC and GMCT*)
CRPc Sends $\{TEK_{i+1}\}$ *Pub − m to m*
Endfor
Endfor Endif

There are two cases of the leaved member: the leaved member is either in transition state so no need to a new key, or in the normal state, so need to change the key in whole group or in its cluster depending on which scheme is used.

4.4. Periodic Rekeying Process

Each cluster has a separate key TEK, so the cluster TEK should be periodically renewed. The CRP generates a new TEK and sends it to all members in the cluster using separate secured channels. For members in the transition state, each one of them will receive this new TEK encrypted by its public key and will no longer use the temporary secured channel with their respective parents. The periodic rekeying process should keep the number of members in the transition state as small as possible; this

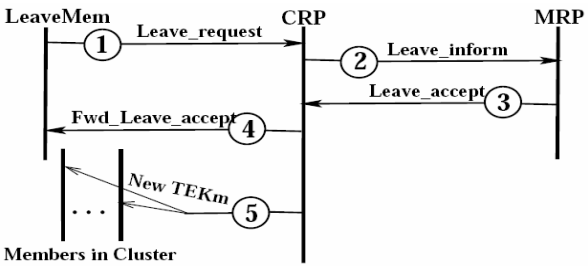

Figure 4. Leaved member with neighbors in the same cluster.

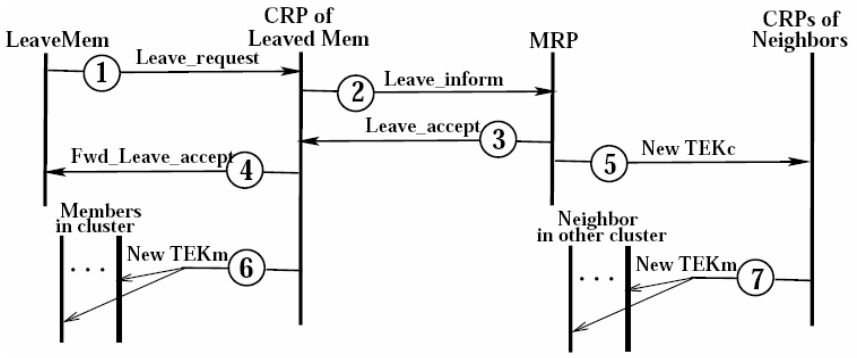

Figure 5. Member leaves, its neighbors in different cluster.

can be done by reducing the rekeying period. However reducing the rekeying period will increases the rekeying overhead. So we should find the right balance between the rekeying period and the average number of members in the transition state depending on the frequency of membership changes in the session. The TCKS re-keying process is illustrated as follows:

Periodic rekeying process steps (Rekeying Process)

Let GMC be the set of all members in cluster C except members in transition state. Let GMCT be all the member in cluster C that are in the transition state. Suppose the current TEK in the Cluster C is TEK_i.

Steps: *For All Clusters c,*

CRPc generates a new key TEK:TEK_{i+1} *for cluster c.*

For all member m ∈ *cluster c* (*GMC and GMCT*)

CRP_c *Sends* {TEK_{i+1}} *Pub −m to m*

Endfor

Endfor

5. Overhead Analysis

The number of cryptographic operations needed in each key management schemes is analyzed. There are asymmetric operations used for key agreement among the group members and symmetric operations used to encrypt the data traffic. We consider the average number of members equal to N, and the session duration is T_s. The data rate is R messages per second, so we can get the number of messages during the session, $N_m = R \times T_s$. The Encryption and decryption times ($N_{Enc/Dec}$) depends on the number of messages. The number of cluster is C, and the average number of members per cluster is NC. Also, the average number of new members is N_n. The results of this analysis are summarized as follows:

	Join	Leave	P. R	$N_{Enc/Dec}$
GKS	O (N)	O (N)	O (N)	$1\times N_m$
CKS	O (N_c)	O (N_c)	O (N)	$C \times N_m$
TKS	1	O (N) or 0	O (N)	$(1 + N_n) \times N_m$
TCKS	1	O (N_c) or 0	O (N)	$(C + N_n) \times N_m$
ATCKS	1	O (N_c) or O(N_n/C)	O (N_c)	$(r \times C) \times N_m$ where $1 < r < 2$

5.1. TEK Management Overhead

We measure the overhead of each scheme in the case of a join, and leave events, and in case of periodic rekeying process as well. **GKS:** *Join*: In order to preserve the backward secrecy, a new TEK should be generated by the MRP and sent to all group members through separate secure channels: this needs N time's asymmetric encryption and decryption operations. *Leave*: In this case a renewal of the TEK is needed to ensure the backward secrecy. So, we need N asymmetric operations. *Periodic rekeying*: When the time to live of the current TEK expires, the MRP triggers a TEK renewal and distribution process. This takes also N asymmetric operations. **CKS:** *Join*: For a join event, the same operations as in the previous scheme (GKS) are needed at the cluster level. The overhead is then N_c asymmetric operations. *Leave*: A leave event needs a local key renewal in the concerned cluster. The overhead is then N_c asymmetric operations. *Periodic rekeying*: Here a complete key renewal is needed in all clusters. The overhead is then N asymmetric operations. **TKS:** *Join*: a new member joins the session; a unique secured channel is established between this new member and its parent in the overlay tree. This transition secured channel will be used until the next general rekeying. The overhead is then one asymmetric operation. *Leave*: In the general case a leave event will triggers a complete rekeying process which takes N asymmetric operations. In some particular cases where the leaving member is still in the transition state, the backward secrecy will be directly ensured without any rekeying, like in the NKS scheme. In this particular case the overhead is null. *Periodic rekeying*: As in all other schemes the general rekeying needs N asymmetric operations. **TCKS:** *Join*: a new member joins the session; a unique secured channel is created between this new member and its parent in the cluster. This transition secured channel will be used until the next rekeying of the cluster. The overhead is then one asymmetric operation as the same as join of TKS. *Leave*: In this case, a leave event will triggers a complete cluster rekeying process which takes N_C asymmetric operations as well not as TKS is N. If the (N/N_C) is not equal to integer value, then there is a cluster having a number of members equals $N_{Cmin} = (N - |N/N_C| \times N_C)$ that is sure less than N_C. Here, the overhead is then less than the overhead of TKS. In some cases, that the leaved member in the transition state, the overhead is null as well as TKS. *Periodic rekeying*: As in all other schemes the general rekeying needs N asymmetric operations. **ATCKS:** *Join*: a new member joins the session; a unique secured channel is created between this new member and its parent in the cluster. This transition secured channel will be used until the next rekeying of the cluster. The overhead is then one asymmetric operation as the same as join of TCKS. *Leave*: In this case, a leave event will triggers a complete cluster rekeying process which takes N_C (mem. in cluster) or N_n/C (mem. in transition state) asymmetric operations as well not as TCKS is N_C only. Here, the overhead is then near the overhead of TCKS. *Periodic rekeying*: As in all other schemes the general rekeying needs N asymmetric operations, but no need to rekeying of this cluster in case of there is no transition member in this cluster. The overhead is then less than the overhead of TCKS.

5.2. Data Encryption/Decryption Overhead

GKS: One key is used for the whole group, so each message is encrypted once at the source side, and decrypted at the member side. So we need $(1 \times N_m)$ symmetric operations. **CKS:** Here there is a key for each cluster. So each message is encrypted once at the source and decrypted/re-encrypted by the cluster GWM at the boundary of each cluster. So we need $(C \times N_m)$ symmetric operations. **TKS:** In TKS scheme, there is a key for all members in the group except for those who are still in the transition state. So each message is encrypted/decrypted once for all members in the group except for members in transition state. For those specific members the message is decrypted/re-encrypted once again by their parents before it is forwarded through the transition secured channel. If we note that average new comers (N_n) for each period during the session, also it means as the average number of members in the transition state. So, we need $((1 + N_n) \times N_m)$ symmetric operations. **TCKS:** In TCKS scheme, there is a key for each cluster except for those who are still in the transition state for each cluster. For all members in clusters except those who are still in the transition state. Each message is encrypted/decrypted at GWMs by C times. For the members in the transition state, we note that the average number of members who are in transition state is N_n. So we need $((C + N_n) \times N_m)$ symmetric operations. **ATCKS:** In ATCKS scheme, there is a key for each cluster except for those who are still in the transition state for each cluster. There is another key for the members who are still in the transition state. Each message is encrypted/decrypted at GWMs by C times. For the members in the transition state, those members are attached to some clusters not all. It is possible that there are clusters without members in transition state. The message encrypted/decrypted in the boundary between the cluster and its transition members by less or equal C times. So we need $(r * C) * N_m)$ symmetric operations, where r = 1… 2. At r = 1, it means that there is no members in transition state and at r = 2, it means all cluster has a member in transition state.

6. Results and Discussion

6.1. Simulation Setup

To measure the security overhead of the four schemes, we used an event-driven simulation of ESM written in C++ under windows 7 on 2.27 GHz intel-core™ i3 processor machine with memory of 4.0 GB. We considered a multicast application with a 256 packet per second rate, with 1 kb packets. This gives an overall data rate of 256 kb/s. The rekeying period is fixed to 80 seconds and the number of clusters for CKS, TCKS and ATCKS schemes is fixed to 5. The mean inter-arrive delay of new members and the mean inter-depart delay are supposed as 20 seconds. We varied the membership duration to get the appropriate mean group size (varied from 20 to 300 members) and the session duration is 55700 seconds.

6.2. Join/Leave/Rekey Overhead Discussion

In order to compare the performance of the five rekeying schemes, we measured the overall security overhead. This includes join, leave and rekeying operations overhead. This overall parameter gives an idea of the efficiency of each scheme in dynamic multicast sessions. Also, taking into account the data encrypt/decrypt, when comparing those schemes. **Figure 6** depicts the security overhead resulting from the join process versus mean group sizes (*i.e.* number of members in the group). It is noted that the TKS, TCKS, and ATCKS has less overhead than both GKS and CKS, because both TKS, TCKS, and ATCKS make a transition secure channel for the new member till rekeying process comes. For both GKS and CKS, the joining process needs to generate a new key for either as the whole group in the GKS or the cluster in the CKS; So GKS has the higher overhead than CKS.

Figure 7 shows the leave overhead versus the mean group size. It is shown that GKS has the higher overhead because when a member leaves the session, a new key have to be generated for the whole group. But in TKS, there are two cases: leaved member either in the transition state so no need to a new key, or in normal member, so need to change the key in the whole group. The TKS has then less overhead than GKS. Also, in the CKS, the new key is generated, because of leaved member, for only the cluster of leaved member. In the TCKS, it likes TKS except in case that leaved member is normal member, so need to generate a new key of a cluster that has this leaved member. TCKS has then less overhead than GKS, TKS, and CKS.

In **Figure 7**, it seems both TCKS and ATCKS have the same leave overhead, but it is not because in the case of

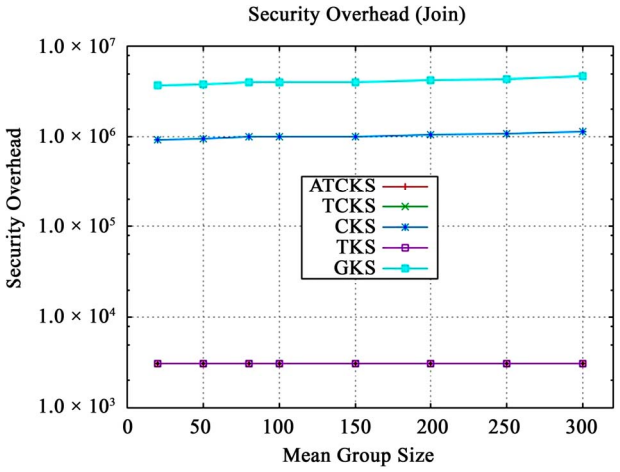

Figure 6. Join security overhead.

leaved member in transition state, a new key have to be generated for the rest transition member of the cluster in which there is a leaved member. **Figure 8**, it is a zooming of a part of the **Figure 7**. It is noted that the leaved member overhead in ATCKS is a little more of that in TCKS, because in the case of leaved member in transition state, for TCKS, no need a new key, but for ATCKS, need a new key for the rest of transition member of the cluster of leaved member. **Figure 9** depicts the rekey overhead versus the mean group size. In the schemes (GKS, TKS, CKS, and TCKS), the keys have to be renewed periodically. So the security overhead of them are the same because in case of using a key as in GKS or TKS, or more than one keys as in CKS or TCKS. But for the ATCKS, not all clusters have a new key. It means that the whole clusters renew the key in different time under condition of the existence of transition members in the cluster. All the schemes have to be renewed periodically against intruder attacks but not in the same time. Because of this, the rekey overheads of ATCKS are reduced. So, the rekey overhead of ATCKS is less than the other schemes. **Figure 10** gives the overall security overhead for the five schemes GKS, TKS, CKS, TCKS, and ATCKS. As expected we notice that our ATCKS scheme gives the best overhead performances.

Figure 11 represents the cumulated overall security overhead for a 300 group members. The security overhead is in log scale versus session time in seconds. We note that ATCKS scheme has the less overall security overhead than GKS, TKS, CKS and TCKS schemes. It is noted that our ATCKS scheme reduces the overall security overhead by 90% - 95% compared to GKS, by 70% - 80% compared to CKS, by 50% - 60% compared to TKS, and by 30% - 40% compared to TCKS.

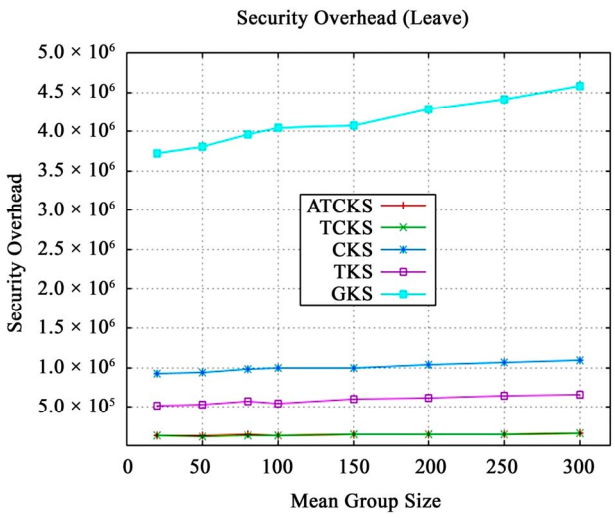

Figure 7. Leave security overhead.

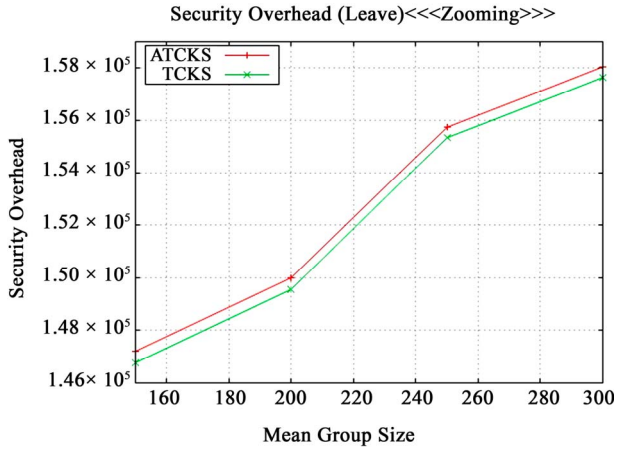

Figure 8. Leave security overhead (zooming).

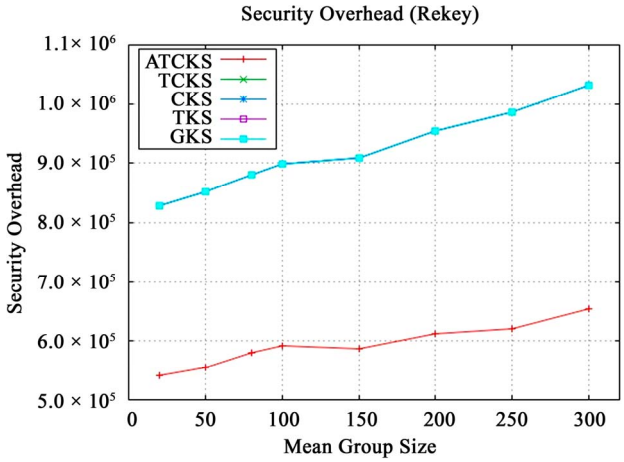

Figure 9. Rekey security overhead.

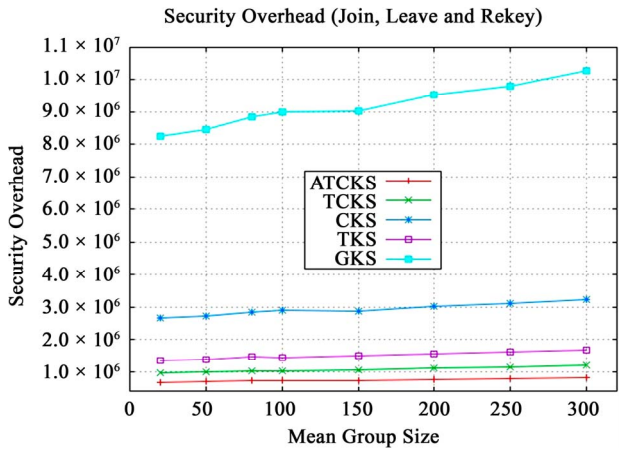

Figure 10. Join/leave/rekey security overhead.

6.3. Data Encrypt/Decrypt Overhead Discussion

The main goal is the security, so we balance between the security performance and control overhead. If we need the system to be more secure, change the key rapidly and taking into account the dynamicity of group members.

Figure 12 depicts the number of encrypt/decrypt operation versus group size. It is noted that more secure, more overhead as in TCKS. With the same security performance as TCKS, ATCKS reduces the security overhead by 6.6%; this is because of some development of TCKS. As shown in **Figure 13**, number of encrypt/decrypt operations of all schemes are compared with that of TCKS that has the highest number of encrypt/decrypt operations and more secure than other schemes except ATCKS. It notes that the number of encrypt/decrypt operations is reduced by approximately 6.6% for ATCKS, 28% for CKS, 55% for TKS, and 85% for GKS.

6.4. Total Overhead Discussion

The number of encrypt/decrypt operations (EncDec) represents the data flow, so we can compare the overhead due to key management, with the number of EncDec operations.

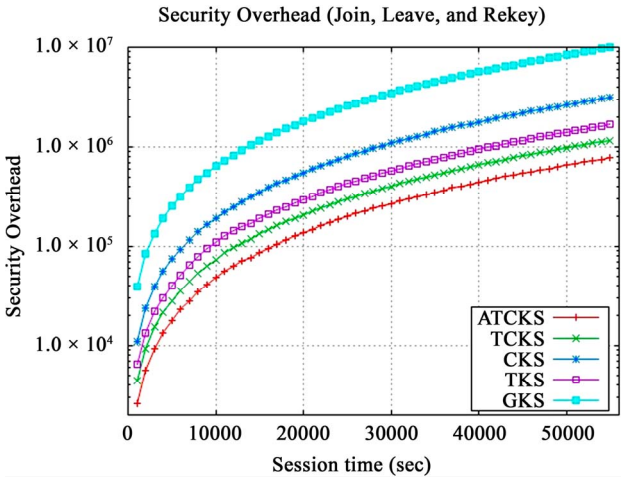

Figure 11. Cumulated overall security overhead for the five schemes and the group size = 300.

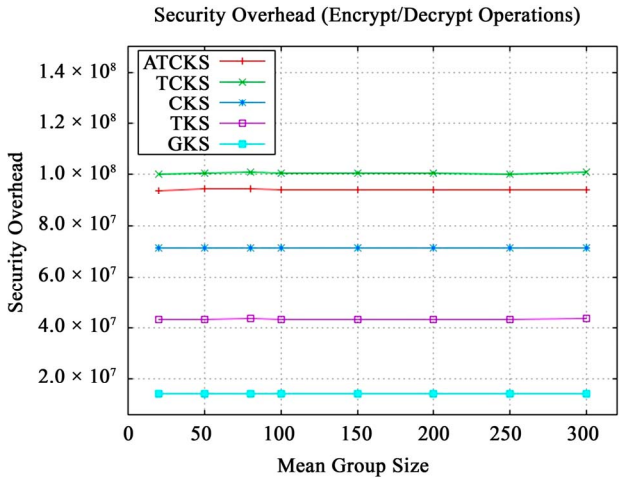

Figure 12. Number of encryption/decryption operation versus group size, for ATCKS, TCKS, CKS, TKS and GKS.

Figure 14 depicts the ratio of control overhead/EncDec operations versus group size. It noted that GKS has the highest ratio more than 50%. The other schemes have the ratio less than 10%. The reasonable ratio of control/data is less that 5%. So, GKS is canceled from the figure to see **Figure 15**. It is noted that ATCKS and TCKS have the minimum ratio less than 1.5% that is a good ratio with more secure as well. Also, TKS and CKS are canceled from the figure, to see the difference between ATCKS and TCKS in **Figure 16**. ATCKS has the ratio (0.7% - 0.9%) that a good ratio with more secure of key management. Finally, ATCKS satisfies less control overhead and more secure for multicast group members.

7. Conclusion

It is very important to reduce bandwidth and protect the packet security during data transmission. In this paper,

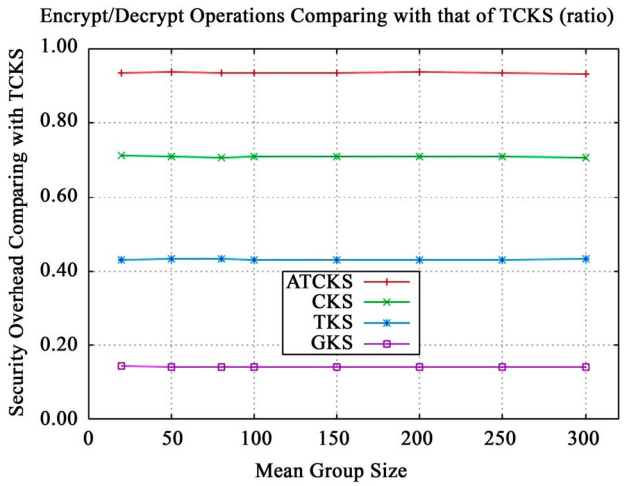

Figure 13. Comparing number of encrypt/decrypt operation for ATCKS, CKS, TKS, GKS with TCKS versus group size.

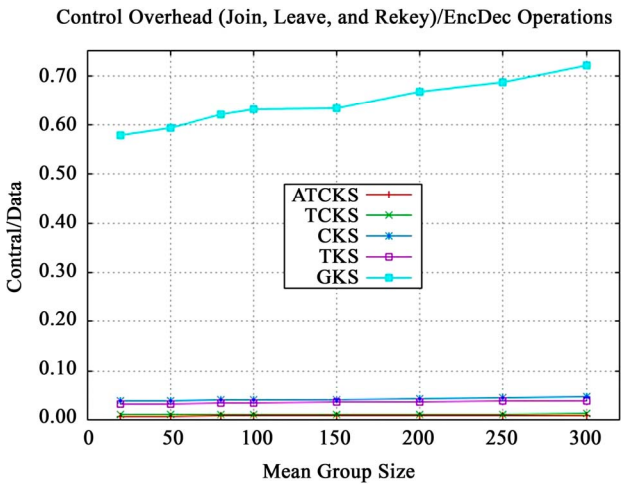

Figure 14. Control overhead over number of encryption/ decryption operation versus group size.

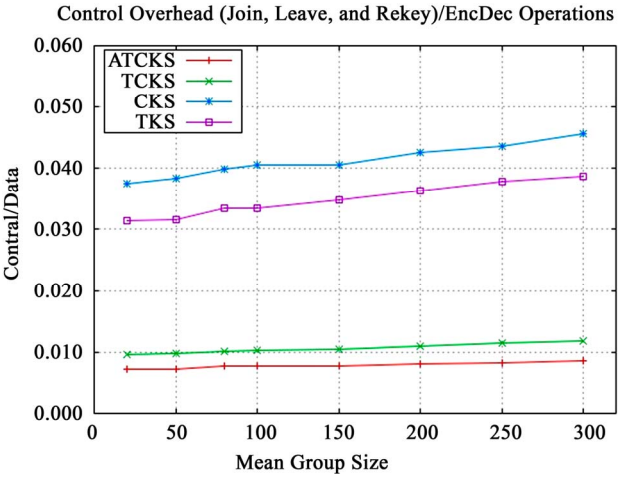

Figure 15. Control overhead over number of encryption/ decryption operation versus group size.

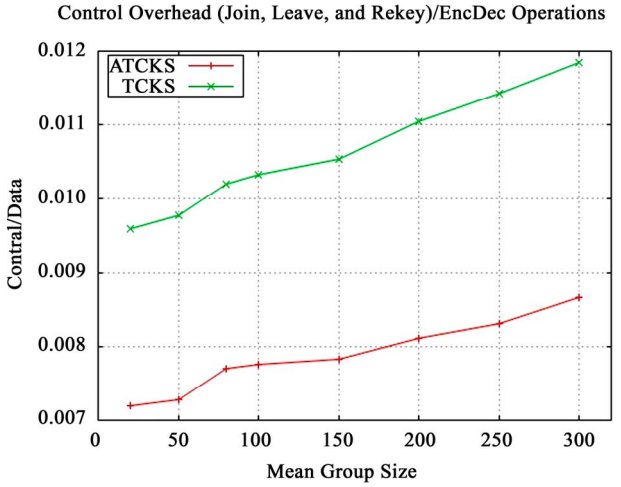

Figure 16. Control overhead over number of encryption/ decryption operation versus group size.

we have studied the different key management schemes and proposed a new scheme namely ATCKS, which is an efficient key management scheme for end-system multicast. Our scheme used a unique TEK for each cluster and a transition key for each cluster to individually manage the joining members to clusters tell a rekeying period and there is no member in transition state, no rekeying done. After the renewal of the TEK in the cluster, these joining members will be fully integrated into its cluster and will receive the new TEK from its CRP. It is shown that our scheme reduces significantly the overall security overhead compared to all other schemes and more reducing the ratio between control overheads and data.

REFERENCES

[1] A. El-Sayed, "A New Approach for Centralized End-System Multicast Protocol," *International Journal of Information Acquisition*, Vol. 3, No. 1, 2006, pp. 77-84.

[2] W. Yiu and S. Chan, "SOT: Secure Overlay Tree for Application-Layer Multicast," *Proceedings of IEEE International Conference on Communications ICC*04, Paris, 20-24 June 2004, pp. 1451-1456.

[3] L. Mathy, R. Canonico and D. Hutchison, "An Overlay Tree Building Control Protocol," *Proceeding of the* 3*rd International COST*264 *Workshop*, Networked Group Communication (NGC 2001), London, 2001, pp. 76-87.

[4] Y.-H. Chu, S. Rao and H. Zhang, "A Case for End System Multicast," *IEEE Journal on Selected Areas in Communication, Special Issue on Networking Support for Multicast*, Vol. 20, No. 8, 2002, pp. 1-12.

[5] A. Chakrabarti and G. Manimaran, "A Case for Mesh-Tree-Interaction in End System Multicasting," *Networking* 2004, LNCS 3042, 2001, pp. 186-199.

[6] V. Roca and A. El-Sayed, "A Host-Based Multicast (HBM) Solution for Group Communications," 1*st IEEE International Conference on Networking*, Colmar, 9-13 July 2001, pp. 610-619.

[7] A. El-Sayed, "Semi-Centralized Approach for End-System Multicast Protocol," *Menoufyia Journal of Faculty of Electronic Engineering Research*, Vol. 15, No. 2, 2005.

[8] M. A. D. Kaafar, T. Turletti and W. Dabbous, "A Locating-First Approach for Scalable Overlay Multicast," 14*th IEEE International Workshop on Quality of Service*, New Haven, 19-21 June 2006, pp. 2-11.

[9] L. Mathy, N. Blundell, A. El-Sayed and V. Roca, "Impact of Simple Cheating in Application-Level Multicast," IEEE Infocom, 2004.

[10] D. Li, Y. Cui, K. Xu and J. Wu, "Impact of Receiver Cheating on the Stability of ALM Tree," *Proceedings of IEEE Global Telecommunications Conference*, St. Louis, Vol. 2, 2005, pp. 667-671.

[11] M. Alkubeily, H. Bettahar and A. Bouabdallah, "Impact of Cheating and Non-Cooperation on the Stability and the Performances of Application Level Multicast Sessions," *Proceedings of the* 4*th International Conference on Information Assurance and Security*, Naples, 8-10 September 2008, pp. 141-146.

[12] M. Alkubeily, H. Bettahar and A. Bouabdallah, "MDA-ALM: Membership Duration Aware Applicationlevel Multicast," *Proceedings of the* 1*st International Global Information Infrastructure Symposium Closing the Digital Divide*, Marrakech, 2-5 July 2007, pp. 120-127.

[13] S. Shetty, P. Galdames, W. Tavanapong and Y. Cai, "Detecting Malicious Peers in Overlay Multicast Streaming," *Proceedings of the* 31*st IEEE International Conference on Local Computer Networks*, Tampa, 14-16 November 2006, pp. 499-506.

[14] A. El-Sayed, "Avoidance Algorithms for an Overlay Multicast Protocol against Cheating," 17*th International Conference on Computer Theory and Applications*, Alexandria, 1-3 September 2007.

[15] H. Ragab, A. Bouabdallah, H. Bettahar and Y. Challal, "Key Management for Content Access Control in a Hierarchy," *International Journal of Computer Networks*, Vol. 51, No. 11, 2007, pp. 3197-3219.

[16] Y. Challal and H. Seba, "Group Key Management Protocols: A Novel Taxonomy," *International Journal of Information Technology*, Vol. 2, No. 1, 2005, pp. 105-118.

[17] H. Bettahar, M. Alkubeily and A. Bouabdallah, "TKS: A Transition Key Management Scheme for Secure Application Level Multicast," *International Journal of Security and Networks*, Vol. 4, No. 4, 2009, pp. 210-222.

[18] F. R. Yu, H. Tang, P. C. Mason, F. Wang, "A Hierarchical Identity Based Key Management Scheme in Tactical Mobile Ad Hoc Networks," *IEEE Transactions on Network and Service Management*, Vol. 7, No. 4, 2010, pp. 258-267.

[19] R. C. Gangwar, "Secure and Efficient Group Key Agreement Protocol for Mobile Ad Hoc Network," *International Journal of Research and Reviews in Computer Science*, Vol. 2, No. 4, 2011.

[20] C. Abad, I. Gupta and W. Yurcik, "Adding Confidentiality to Application-Level Multicast by Leveraging the Multicast Overlay," *Proceedings of the* 4*th International Workshop on Assurance in Distributed System and Networks*, Columbus, 2-4 February 2005, pp. 5-11.

[21] A. El-Sayed, "A New Secure Group Management and Communication in End-System Multicast Protocol," *International Journal of Computer Science & Network Security*, Vol. 10, No. 3, 2010, pp. 302-310.

